Subject: sample/empirical variogram calculation
Posted by james-a-roo on Wed, 29 Oct 2008 20:01:30 GMT
View Forum Message <> Reply to Message

Hi all-

I've decided that I want to be able to compute the empirical variogram in IDL. More specifically i want to compute the semivariance in 2D and be able to specify arbitrary bins in distance and and (even number of) arbitrary bins in direction.

There was an old post on this, but the code wasnt really satsifactory for my purposes. I've looked through several folks' libraries and havent turned up anything. I thought I'd ask here before getting to far into it (though i'm already part way there).

Any one have a routine for directional, empirical variograms in IDL?

Thanks

Subject: Re: sample/empirical variogram calculation
Posted by james-a-roo on Thu, 06 Nov 2008 05:54:24 GMT
View Forum Message <> Reply to Message

Even though this posting generated zero interest in the last week, there were two old posts on the topic and I have a pet peeve when people don't post their answers to their questions. So here's the code i wrote. I'll make another post following this one with a few crude examples. I'm fairly confident I have the results correct, though I havent run numbers against any other routine, say in R.

Right now, this routine accepts 1-3 dimensional data. Bins in each dimension are fairly flexibly specified either as regular or irregular bins. Regular bins are efficiently processed directly by JD Smith's hist_nd, irregular bins are transformed to regular indices via value_locate before histogramming. I think there's just no way around the looping nature of this beast. I first wrote a code to avoid that, but it wont function for input arrays over even a modest size as it constructs several nxn matrices from n input points. The most obvious drawback with this currently is that angle bins (spherical coords using theta and phi) are not permitted to be rotated relative to the data. I think this should be really easy to implement, just add or subtract the desired angle from the data once it's transformed to spherical coordinates, but I havent done that and dont have the time to check it for those purposes right now.

The main concern with this code is improving the efficiency of what's happening inside the loop, as the number of laps is potentially high. I'm not the best versed in what kind of array manipulations and declarations are fastest and most memory efficient. So there's probably some easy room for speed-up. I also didnt see any way to calculate the largest necessary bin in radius in advance of the loop, these calculations slow the looping in the case that the largest bin bounds in radius are not supplied.

I hope someone gets use out of this! If you are seeing any highly desirable feautres that appear lacking, do post or email me to see if I may have already added them, esp if some time has gone by.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;; ;;;;;;;;;;;;;;;;;;;


; NAME: variogram
;
; PURPOSE:
; Calculate the EMPIRICAL variogram of a large data set in 1-3
dimensions for arbitrary
; distance bins. If 2D, also
; an arbitrary number of bins in [0,180) in the x-y plane. If 3D, also
an arbitrary number of
; bins in the azimuthal direction.
; Note: 3D is not fully implemented as of right now.
; Note: The euclidean norm is currently the measure of distance
between 2 points.
;
; CATEGORY: spatial stats
;
; CALLING SEQUENCE:
;
; INPUTS:
; x - cartesian, primary dimension of location of observations, length
nobs
; y - carteisan, secondary dimsension of location of obs, length nobs.
; z - carteisan, tertiary dimension of location of obs, length nobs.
; v - the value of the observation at the corresponding location,
length nobs.

; OPTIONAL INPUTS:
; radiusbins -
;   Default or no input returns [0,maxrad+fudge,1] where 0 is the
start of the smallest bin and 1
;   is the radius bin width. Maxrad+fudge is the lower bound of the
last bin which contains the
;   largest radius found i nthe data.
;   A scalar implies the binwidth wich is applied from 0 to include
```

```
the largest radius found
;   in the data. In this case, [0,maxrad+fudge,binwidth] is returned.
;   A three element vector [min,max,binwidth] specifies the lower
(inclusive) bounds of the first and
;   last bin and the width of the bins intervening, which requires
that (max-min) is an integer.
;   Any other number of elements implies the bin starts and ends and
may be completely irregular.
;   Each bin start is included in that bin and the final value is also
included in the final bin.
;   In this last case, elements must be positive and in monotonically
increasing order. Example:
;   eg: [0,1,!pi,8] -> bins: [0,1), [1,!pi), [!pi,8]
; thetabins - For 2D or 3D data.
;   Undefined implies 1 bin, isotropic calcualtion.
;   Scalar value implies the number of bins in [0,!pi).
;   More than 1 element imples bin boundaries. Elements must be
positive,monotincally increasing,
;   and in the interval [-!pi/2,!pi/2). If the first and last values
are not
;   -!pi/2 and !pi/2, respectively, then these are appended to the
approporiate ends of the array.
;   The returned value is the list of lower bounds (inclusive) on each
bin starting with 0 and
;   going through the !pi/2.
; phibins - For 3D data.
;   Same as thetabins but for 3D data. This is the azimuthal direction
(as measured from
;   the z axis).
;   Example: [45,90,135]*!pi/180. -> [0,45,90,135,180]*!pi/180 ->
;               [0,45), [45,90), [90,135), [135,180) all *!pi/180.
; subsetinds - By default the empirical variogram is calculated as the
expected value in each bin
;   (bins are defined by the spatial partioning of the above 3
parameters) over ALL
;   supplied observations. If the expectation is only desired over
some subset of input points,
;   these are the indices of these observation n-tuplets corresponding
to the location in the
;   observation vectors x, (y, (z)), v.
; double - Indicates calculations to be done in double precision. As
histogram is a "razor's" edge,
;   inputs need to be in double precision for this to work. A message
will alert the user if this is
;   not the case for any inputs involved in the calculation.
; semi - Returns the semivariogram=variogram/2 (=variance of a
stationary random field w/o spatial dependence).
; KEYWORD PARAMETERS:
```

```
;
; OUTPUTS:
; return = [nradiusbins] , [nradiusbins,nthetabins],
[nradiusbins,nthetabins,nphibins]
;   depending on dimensions of input.
; OPTIONAL OUTPUTS:
;
; COMMON BLOCKS:
;
; EXTERNAL CALLS:
;
; SIDE EFFECTS:
;
; RESTRICTIONS:
;
; PROCEDURE:
;
; EXAMPLE:
;   x=[0,1,1,0,.2,1,0,-.2,1.8]
;   y=[0,0,1,1,.2,-1,-1,-.2,0]
;   v=[0,1,1,1,.1,3,3,-.2,1.6]
;   radiusbins=.8
;   thetabins=2
;
 r=variogram_big(x=x,y=y,radiusbins=radiusbins,thetabins=thet abins,v=v);;
;   more examples are posted with the code on the google groups site
;   http://groups.google.com/group/comp.lang.idl-pvwave
;
; REFERNCES: see wikipedia: http://en.wikipedia.org/wiki/Variogram
;
; MODIFICATION HISTORY:
; 10/30/08 - jlm
; Please post improvement, corrections, modifications back to the
google page from whence this may have come.
;
 ;********************************************************************* *******************************
; Use this code entirely at your own risk. No warranties of any sort,
express, implied,
; or infered regarding anything are offered, including software,
hardware, user sanity.
; Nothing is offered except the code.
; The only requirement of use or reproduction is that you keep the
author credits in
; tact and contribute to the improvement when possible.
 ;********************************************************************* *******************************
;-
function variogram, x=x, y=y, z=z, v=v, semi=semi,  $
                        radiusbins=radiusbins, thetabins=thetabins,
phibins=phibins, $
                        subset_inds=subset_inds, double=double,
cumulative_hist=cumulative_hist

 nx=n_elements(x)                        ; x coordinates
 nv=n_elements(v)                        ; value at coordinates
 IF keyword_set(y) THEN ny=n_elements(y) ; y coordinates
 IF keyword_set(z) THEN nz=n_elements(z) ; y coordinates
 message=0
 IF nx NE nv THEN message=1
 IF keyword_set(y) THEN IF nx NE ny THEN message=1
 IF keyword_set(z) THEN IF nx NE nz THEN message=1
 IF message THEN message,'Input data vectors must have same lengths'
 nobs=nx
 undefine,nx,ny,nz,nv

 ;; Type defaults to float if double is not set.
 type = keyword_set(double) OR size(v,/type) EQ 5 ? 5L : 4L
 badtypes=0
 IF size(x,/type) NE type THEN badtypes=badtypes+1
 IF keyword_set(y) THEN IF size(x,/type) NE type THEN badtypes=badtypes
+1
 IF keyword_set(z) THEN IF size(x,/type) NE type THEN badtypes=badtypes
+1
 IF keyword_set(v) THEN IF size(x,/type) NE type THEN badtypes=badtypes
+1
 IF keyword_set(radiusbins) THEN IF size(x,/type) NE type THEN
badtypes=badtypes+1
 IF keyword_set(thetabins) THEN IF size(x,/type) NE type THEN
badtypes=badtypes+1
 IF keyword_set(phibins) THEN IF size(x,/type) NE type THEN
badtypes=badtypes+1
 IF badtypes GT 0 THEN message,'There are '+string(badtypes,'(I1.1)')
+' input variables of the wrong type.'

 undefine,cumulative_hist

 zero = type EQ 4 ? 0. : 0.D
 one = type EQ 4 ? 1. : 1.D
 negone = type EQ 4 ? -1. : -1.D
 two = type EQ 4 ? 2. : 2.D
 pi = type EQ 4 ? !pi : !dpi
 pi2 = type EQ 4 ? !pi/2. : !dpi/2.D
 precision=(machar(double=type-4)).eps

 ;; Treat the radiusbins keyword.
 ;; Note that we dont necessarily know the number of bins in radius a
priori.
```

```
 IF ~keyword_set(radiusbins) OR n_elements(radiusbins) EQ 1 THEN
BEGIN
   radius_bounds_specified=0
   radiusmin=zero
   radiusmax=zero ;; must initialize this less than any possible
maximum radius
   ;; If no bin iformation was specified, use binwidth of 1
   radiusbinwidth= ~keyword_set(radiusbins) ? one : radiusbins
   radius_regular=1
 ENDIF ELSE BEGIN
   radius_bounds_specified=1
   radiusmin=radiusbins[0]
   radiusmax=radiusbins[n_elements(radiusbins)-1]
;   radiusbins = radiusbins
   nradiusbins=n_elements(radiusbins)-1
   ;; Check if the passed array is evenly spaced
   radius_regular = $
     min( (radiusbins[1]-radiusbins[0])-(radiusbins[1:nradiusbins]-
radiusbins[0:nradiusbins-1]) LT precision )
   IF radius_regular THEN radiusbinwidth = radiusbins[1]-radiusbins[0]
ELSE BEGIN
     message, 'Bin spacing in the radius dimension is irregular. This
will greatly slow the variogram '+$
              'calculation. Consider a transform beforehand to regular
spacing, if possible.',/info
     ;; If irregular, there will be a "coordinate" transform on the
data which will give the follwing bounds
     radiusmin=zero
     radiusmax=nradiusbins
     radiusbinwidth=one
   ENDELSE
 ENDELSE
 IF radiusmin GT 0 THEN message,'*** Warning: The lowest radius bin is
greater than zero',/info

 ;; If 2D or 3D get the xy plane bins, thetabins
 IF keyword_set(y) THEN BEGIN
   thetamin=negone*pi2
   thetamax=pi2
   IF ~keyword_set(thetabins) THEN BEGIN
     nthetabins=1
     thetabinwidth=pi
     thetabins= type EQ 4 ? findgen(2)*thetabinwidth-pi2 :
dindgen(2)*thetabinwidth-pi2
     theta_regular=1
   ENDIF ELSE BEGIN
     IF n_elements(thetabins) EQ 1 THEN BEGIN
       nthetabins=fix(ceil(thetabins[0]),type=type)  ;; make it an
```

```
"integer" in case it's not already.
       thetabinwidth=pi/nthetabins
       thetabins = type EQ 4 ? findgen(nthetabins+1)*thetabinwidth-
pi2 : dindgen(nthetabins+1)*thetabinwidth-pi2
       theta_regular=1
     ENDIF ELSE BEGIN
       ;; check that bins are within bounds and monotonically
increasing.
       IF max(abs(thetabins)) GT pi2 OR max(sort(thetabins)-
indgen(n_elements(thetabins))) NE 0 THEN $
         message,'*** Warning: Supplied thetabins is not of the
required form.'
       IF abs(thetabins[0]) NE pi2 THEN thetabins=[negone*pi2,
thetabins]
       IF thetabins[n_elements(thetabins)-1] NE pi2 THEN
thetabins=[thetabins,pi2]
       nthetabins=n_elements(thetabins)-1
       ;; Check if the passed array was evenly spaced
       theta_regular = $
         min( abs((thetabins[1]-thetabins[0])-(thetabins[1:nthetabins]-
thetabins[0:nthetabins-1])) LE precision )
       IF theta_regular THEN thetabinwidth = thetabins[1]-thetabins[0]
ELSE BEGIN
         message, 'Bin spacing in the theta dimension is irregular.
This will greatly slow the variogram '+$
                  'calculation. Consider a transform beforehand to
regular spacing, if possible.',/info
         ;; If irregular, there will be a "coordinate" transform on
the data which will give the follwing bounds
         thetamin=zero
         thetamax=nthetabins
         thetabinwidth=one
       ENDELSE
     ENDELSE
   ENDELSE
 ENDIF

 ;; If 3D get the azimuthal bins, phibins
 IF keyword_set(z) THEN BEGIN
   phimin=zero
   phimax=pi
   IF ~keyword_set(phibins) THEN BEGIN
     nphibins=1
     phibinwidth=fix(pi,type=type)
     phibins= type EQ 4 ? findgen( 2 )*thetabinwidth :
dindgen( 2 )*thetabinwidth
     phi_regular=1
   ENDIF ELSE BEGIN
```

```
     IF n_elements(phibins) EQ 1 THEN BEGIN
       nphibins=fix(ceil(phibins[0]),type=type)  ;; make it an integer
in case it's not already.
       phibinwidth=pi/phibins
       phibins = type EQ 4 ? findgen(nphibins+1)*phibinwidth :
dindgen(nphibins+1)*phibinwidth
       phi_regular=1
     ENDIF ELSE BEGIN
       ;; check that bins are within bounds and monontonically
increasing
       IF min(phibins) LT 0 OR max(phibins) GT pi OR $
         max(sort(phibins)-indgen(n_elements(phibins))) NE 0 THEN $
           message,'*** Warning: Supplied phibins is not of the
required form.'
       IF phibins[0] NE zero THEN phibins=[zero,phibins]
       IF phibins[n_elements(phibins)-1] NE pi THEN
phibins=[phibins,pi]
       nphibins=n_elements(phibins)-1
       ;; Check if the passed array was evenly spaced
       phi_regular = min( abs((phibins[1]-phibins[0])-
(phibins[1:nphibins]-phibins[0:nphibins-1])) LE precision )
       IF phi_regular THEN phibinwidth = phibins[1]-phibins[0] ELSE
BEGIN
         ;; If irregular, there will be a "coordinate" transform on
the data which will give the follwing bounds
         message, 'Bin spacing in the phi dimension is irregular. This
will greatly slow the variogram '+$
                  'calculation. Consider a transform beforehand to
regular spacing, if possible.',/info
         phimin=zero
         phimax=nphibins
         phibinwidth=one
       ENDELSE
     ENDELSE
   ENDELSE
 ENDIF

 ;; Set up parameters outside the loop.
 ;; The following number (inspired by chmod) describes which
dimensions have regular bins.
 bins_regular=0
 IF radius_regular THEN bins_regular=bins_regular+1
 IF keyword_set(y) THEN IF theta_regular THEN bins_regular=bins_regular
+2
 IF keyword_set(z) THEN IF phi_regular THEN bins_regular=bins_regular
+4
 ;; Flag here if a certain dimension needs a coordinate transform,
reduce the # of ops in the loop.
```

```
 radius_xform = max(bins_regular EQ [0, 2, 4, 6])  ; 1 if any is true
0 if none are <-> OR
 theta_xform = keyword_set(y) ? max(bins_regular EQ [0, 1, 4, 5]) : 0
 phi_xform = keyword_set(z) ? max(bins_regular EQ  [0, 1, 2, 3]) : 0
 ;; Might as well.
 ndims= keyword_set(x) + keyword_set(y) + keyword_set(z)

 CASE ndims OF
   1: BEGIN
     min = [radiusmin]
     max = [radiusmax]  ;; radiusmax may be changing below
     binsize = [radiusbinwidth]
   END
   2: BEGIN
     min = [radiusmin, thetamin]
     max = [radiusmax, thetamax]  ;; radiusmax may be changing below
     binsize = [radiusbinwidth, thetabinwidth]
   END
   3: BEGIN
     min = [radiusmin, thetamin, phimin]
     max = [radiusmax, thetamax, phimax]  ;; radiusmax may be changing
below
     binsize = [radiusbinwidth, thetabinwidth, phibinwidth]
   END
 ENDCASE

 IF ~keyword_set(subset_inds) THEN subset_inds=lindgen(nobs) ELSE $
   IF max(subset_inds) GE nobs THEN message,'The passed subset_inds is
out of range of the input data.'
 nss_inds=n_elements(subset_inds)

 ;; loop over all points
 FOR ss=0L,nss_inds-1 DO BEGIN

   xdists = x-x[ss]
   ydists = keyword_set(y) ? y-y[ss] : zero ;these are just dummies if
not defined.
   zdists = keyword_set(z) ? z-z[ss] : zero

   ;; *** Go to spherical coords: radius, theta, phi ***
   ;; Get the (euclidean) vector distances/radii
   radius=sqrt( xdists^two + ydists^two + zdists^two )
   ;; Get the theta vector directions if 2D
   IF keyword_set(y) THEN BEGIN
     yonx=temporary(ydists)/temporary(xdists)
     theta=atan( yonx )  ;;  result in [-pi/2,pi/2] rads (crazy!)
     ;; When the divisor is zero, the point lies along 0 radians
(straight up or down).
```

```
    wh0rad=where( finite(yonx) EQ 0 )
    IF wh0rad[0] NE -1 THEN theta[wh0rad]= negone*pi2 ;; want pts in
[-pi/2,pi/2) rads
  ENDIF
  ;; Get the phi (azimuthal) vector directions if 3D
  IF keyword_set(z) THEN BEGIN
    phi=acos( zdists/radius )                ;; result in [0, pi]
rads
    wh0rad=where( finite(phi) EQ 0 )
    IF wh0rad[0] NE -1 THEN phi[wh0rad]= pi2 ;; dosent really matter
where this goes, rad=0 is a spec case.
    phi= phi MOD pi                          ;; result in [0, pi)
rads
  ENDIF

  ;; *** Convert any data with irregular bins to regular bins ***
  ;; This is the afore mentioned "coordinate" transform, bins
boundaries have already been adjusted.
  IF radius_xform THEN radius=value_locate( radiusbins, radius )
  IF theta_xform THEN theta=value_locate( thetabins, theta )
  IF phi_xform THEN phi=value_locate( phibins, phi )

  ;; Assemble data
  CASE ndims OF
    1: histin = transpose(radius)
    2: histin = [ transpose(radius), transpose(theta) ]
    3: histin = [ transpose(radius), transpose(theta),
transpose(phi) ]
  ENDCASE

  ;; If a larger radius is found
  IF max(radius) GT max[0] THEN BEGIN
    IF radius_bounds_specified THEN BEGIN
      IF max(radiusbins) LT max(radius) THEN $
        message,'*** Warning: Supplied radiusbins do not include all
distances found in the data.',/info
    ENDIF ELSE BEGIN
      ;; If the radius bounds arent fixed, we have to grow smaller
cumulative_histo and variogram
      ;; to accomodate
      newnbins=fix(ceil(max(radius)/radiusbinwidth),type=type)
      oldnbins=fix(ceil(max[0]/radiusbinwidth),type=type)
      nnewbins=newnbins-oldnbins
      max[0]= fix(newnbins,type=type)*radiusbinwidth
      CASE ndims OF
        1: add_dims=[nnewbins+1]
        2: add_dims=[nnewbins+1,nthetabins+1]
        3: add_dims=[nnewbins+1,nthetabins+1,nphibins+1]
```

```
       ENDCASE
       cumulative_hist= n_elements(cumulative_hist) EQ 0  ?
make_array( add_dims, type=3) : $
                        [cumulative_hist, make_array( add_dims,
type=3) ]
       variogram=n_elements(variogram) EQ 0  ? make_array( add_dims,
type=type) : $
                 [variogram, make_array( add_dims, type=type) ]
     ENDELSE
   ENDIF

   ;; *** Histogram! Thanks JD. ***
   hist=hist_nd( histin, binsize, min=min, max=max, rev=ri )
   ;; Accumulate histogram
   IF n_elements(cumulative_hist) EQ 0 THEN cumulative_hist=hist ELSE
$
     cumulative_hist=cumulative_hist+hist

   ;; *** Calculate the squared difference of the values ***
   vdists2=(v-v[ss])^two
   ;; Put the vdists2 in the right bins with reverse indices
   variogramtemp = fix(hist*0.,type=type)
   whistne0=where(hist NE 0)
   FOR ww=0,n_elements(whistne0)-1 DO $
     variogramtemp[whistne0[ww]] =
total(vdists2[ ri[ ri[whistne0[ww]]:ri[whistne0[ww]+1]-1 ] ])
   ;; This is really a cumulative vdists2 in each bin. Outside the
loop, the expected value
   ;; is found by dividing by the cumulative_hist.
   variogram = n_elements(variogram) EQ 0 ? variogramtemp : variogram
+variogramtemp

   ;; If there is a bin containing 0 radius, we need to double count
every pt-to-self vdist2=0 (2)
   ;; in ALL directional bins (1).
   ;; 1) We need this over all bins for some continuity in the nugget
over directions. For instance,
   ;;    if the bins contained only pts with zero spatial separation,
there really is no difference
   ;;    in direction.
   ;; 2) Because symmetry between distinct points is double counting
all pt-to-pt sq differences while
   ;;    only single counting pt-to-self sq differences, we need to
double count pt-to-self sq
   ;;    differeneces. Think of the variance of the process and the
nugget effect, if there
   ;;    are points at the same location with positive sqdifferences
(vdist2) then these will be
```

```
   ;;    weighted twice as strongly in the nugget that point-to-self
zeros.
   IF min[0] EQ 0 THEN BEGIN
     CASE ndims OF
       1: cumulative_hist[0]=cumulative_hist[0]+1
       2: cumulative_hist[0,*]=cumulative_hist[0,*]+1
       3: cumulative_hist[0,*,*]=cumulative_hist[0,*,*]+1
     ENDCASE
   ENDIF

 ENDFOR ;; end looping over the subset of obs to accumulate the
variogram and histogram.

 ;; Calculate the variogram  - the mean/expected value in each bin.
 ;; If a bin has no member, this returns a nan, which is desirable
IMO.
 variogram= variogram / fix(cumulative_hist,type=type)
 IF keyword_set(semi) THEN variogram=variogram/two

 ;; Trim off these end point bins in the likely or certain event they
are empty
 size=size(variogram,/dim)
 CASE ndims OF
   1: IF ~max( finite(variogram[size[0]-1]) ) THEN BEGIN
     variogram=variogram[0:size[0]-2] ;; likely
     cumulative_hist=cumulative_hist[0:size[0]-2]
   ENDIF
   2: BEGIN
     IF ~max( finite(variogram[size[0]-1,*]) ) THEN BEGIN
       variogram=variogram[0:size[0]-2,*] ;; likely
       cumulative_hist=cumulative_hist[0:size[0]-2,*]
     ENDIF
     variogram=variogram[*,0:size[1]-2] ;; by construction of the
angle bins, pi/2 -> -pi/2
     cumulative_hist=cumulative_hist[*,0:size[1]-2]
   END
   3:  BEGIN
     IF ~max( finite(variogram[size[0]-1,*,*]) ) THEN BEGIN
       variogram=variogram[0:size[0]-2,*,*] ;; likely
       cumulative_hist=cumulative_hist[0:size[0]-2,*,*]
     ENDIF
     variogram=variogram[*,0:size[1]-2,*] ;; again, pi/2 -> -pi/2
     cumulative_hist=cumulative_hist[*,0:size[1]-2,*]
     variogram=variogram[*,*,0:size[2]-2] ;; here, by construction, pi
-> 0
     cumulative_hist=cumulative_hist[*,*,0:size[2]-2]
   END
 ENDCASE
 ;; Create the output for radius bins
 ;; If radius is on an irregular grid, then radius_bounds_specified=1
and this wont happen as it's
 ;; already set. No need to worry about the "coordinate xform" bounds.
 IF ~radius_bounds_specified THEN $
   radiusbins= findgen( (max[0]-min[0])/radiusbinwidth +1 ) *
radiusbinwidth + min[0]

 return,variogram


END
```

Subject: Re: sample/empirical variogram calculation
Posted by james-a-roo on Thu, 06 Nov 2008 05:58:59 GMT
View Forum Message <> Reply to Message

hmmm... that dosent post very well. I'd provide a link but i'm
currently without a place to really post the code.

here are the rough examples. at least these may be useful to someone
doing some 4am hacking.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;; ************************************************************ ****
;; code supplied as is. use at own risk, no warranties expressed,
;; implied, or infered.
;; ************************************************************ ***

!p.charsize=1.5

IF 0 THEN BEGIN

   x=[0,1,1,0,.2, 1, 0,-.2,1.60001]
   y=[0,0,1,1,.2,-1,-1,-.2,  0    ]
   v=[0,1,1,1,.1, 3, 3,-.2,1.6    ]
   loadct,39,/silent
   plot,x,y,color=0,psym=2
   loadct,33,/silent
   plots,x,y,color=bytscl(v),psym=2,thick=5
   loadct,39

   radiusbins=.8D
   thetabins=2D
```

```
    r=variogram(x=x,y=y,radiusbins=radiusbins,thetabins=thetabin s,v=v,/
double)

   ;;now bins are being passed in from above.
    r2=variogram(x=x,y=y,radiusbins=radiusbins,thetabins=thetabi ns,v=v,/
double)

   phibins=2

 r3=variogram(x=x,y=y,radiusbins=radiusbins,thetabins=thetabi ns,phibins=phibins,v=v,/
double)

   z=randomu(seed,9)

 r4=variogram(x=x,y=y,z=z,radiusbins=radiusbins,thetabins=the tabins,phibins=phibins,v=v,/
double)

 ENDIF

IF 1 THEN BEGIN

   nside=72L
   smoothwidth=15
   x=double(floor(findgen(nside,nside)/nside))
   y=reform(transpose(x),[nside^2])
   x=reform(x,[nside^2])

   ; z=reform(findgen(nside,nside),[nside^2])
   v=randomn(seed,nside,nside)
   window,0,xs=1200,ys=800
   !p.multi=[0,3,2]
   tvscale,v,/nointerp
   !p.multi[0]=!p.multi[0]-2
   v=reform(v,[nside^2])
   radiusbins=5.D  ;width
   thetabins=3.D    ;number of divisions

 r=variogram(x=x,y=y,radiusbins=radiusbins,thetabins=thetabin s,cum=cumu,
v=v,/double)
   nbins=n_elements(radiusbins)-2
   binstarts=radiusbins[0:nbins]
   plot,binstarts,r[*,0],yr=[0,max(r)],/ys
   FOR d=0,n_elements(thetabins)-2 DO
oplot,binstarts,r[*,d],color=d*254/2., thick=2
   !p.multi[0]=!p.multi[0]+3
;   plot,cumu[*,0]
;   FOR d=0,n_elements(thetabins)-2 DO oplot,cumu[*,d],color=d*50.,
thick=2
```

```
   ;; copy the original data
   v0=reform(v,[nside,nside])

   v=smooth(v0,floor(smoothwidth*[1.,1./smoothwidth]),/edge) ; smooth
anisotropically
   tvscale,reform(v),/nointerp
   !p.multi[0]=!p.multi[0]-2
   v=reform(v,[nside^2])

 r=variogram(x=x,y=y,radiusbins=radiusbins,thetabins=thetabin s,cum=cumu,
v=v,/double)
   nbins=n_elements(radiusbins)-2
   binstarts=radiusbins[0:nbins]
   plot,binstarts,r[*,0],yr=[0,max(r)],/ys
   FOR d=0,n_elements(thetabins)-2 DO
oplot,binstarts,r[*,d],color=d*254/2., thick=2
   !p.multi[0]=!p.multi[0]+3

   v=smooth(v0,floor(smoothwidth),/edge)  ;isotropic
   tvscale,reform(v),/nointerp
   !p.multi[0]=!p.multi[0]-2
   v=reform(v,[nside^2])

 r=variogram(x=x,y=y,radiusbins=radiusbins,thetabins=thetabin s,cum=cumu,
v=v,/double)
   nbins=n_elements(radiusbins)-2
   binstarts=radiusbins[0:nbins]
   plot,binstarts,r[*,0],yr=[0,max(r)],/ys
   FOR d=0,n_elements(thetabins)-2 DO
oplot,binstarts,r[*,d],color=d*254/2., thick=2

 ENDIF

IF 0 THEN BEGIN

  nside=50^2
  x=randomu(seed,nside)-.5
  y=randomu(seed,nside)-.5
  z=randomu(seed,nside)-.5
  v=sqrt( x^2. + y^2. + z^2. )

;  x=x[sort(v)]
;  y=y[sort(v)]
;;  z=z[sort(v)]
;  v=v[sort(v)]

  loadct,39,/silent
  window,1,xs=800,ys=800
  SURFACE, DIST(5), /NODATA, /SAVE, XRANGE=[0,1]-.5, $  ;thanks David,
the RSI examples are worthless.
         YRANGE=[0,1]-.5, ZRANGE=[0, 1]-.5, XSTYLE=1, $
         YSTYLE=1, ZSTYLE=1, CHARSIZE=1.5, $
         POSITION=[0.1, 0.1, 0.95, 0.95, 0.1, 0.95], $
         XTICKLEN=1, YTICKLEN=1, XGRIDSTYLE=1, YGRIDSTYLE=1
  AXIS, XAXIS=1, /T3D, CHARSIZE=1.5
  AXIS, YAXIS=1, /T3D, CHARSIZE=1.5

  phi = Findgen(32) * (!PI * 2 / 32.)
  phi = [ phi, phi(0) ]
  UserSym, Cos(phi), Sin(phi), /Fill
  loadct,33,/silent
  PLOTS, x, y, z, PSYM=8, COLOR=bytscl(v), SYMSIZE=2.5, /T3D
  plots, [0,0],[0,0],[-1,1]*.5,/t3d,thick=2
  plots, [0,0],[-1,1]*.5,[0,0],/t3d,thick=2
  plots, [-1,1]*.5,[0,0],[0,0],/t3d,thick=2
;  FOR pp=0,nside-1 DO plots,[0,x[pp]],[0,y[pp]],[0,z[pp]],/t3d
  loadct,39,/silent

  radiusbins=.05
  thetabins=[-1D*!dpi/2.D, .1D, !dpi/2.D]  ;2  ;trying regular and
irregularly specified bins
  phibins=[0.D, (!dpi/(2.D))-1D, !dpi]  ;2
  r=variogram(x=x,y=y,z=z,$

radiusbins=radiusbins,thetabins=thetabins,phibins=phibins,$
                cum=cumu, v=v,/double)
  nbins=n_elements(radiusbins)-2
  binstarts=radiusbins[0:nbins]
  window,2,xs=800,ys=800
  plot,binstarts,r[*,0,0],yr=[0,max(r)]
  FOR t=0,1 DO FOR p=0,1 DO
oplot,binstarts,r[*,t,p],color=((t*2)+p)*254/3.


ENDIF

END
```

Subject: Re: sample/empirical variogram calculation
Posted by matsu770610 on Fri, 07 Nov 2008 03:26:38 GMT

View Forum Message <> Reply to Message

i am searching for a IDL library of geostatistics, but it seems that
people using IDL doesnt concern this subject. R is perfect for this

subject, but is too slow and has memory problems when imported large dataset.

Subject: Re: sample/empirical variogram calculation
Posted by matsu770610 on Fri, 07 Nov 2008 06:59:02 GMT

View Forum Message <> Reply to Message

Maybe Gslib library is a possible choise that can be linked to IDL to perform some kind of geo-statistics calculation. However, it's fortran code library so i have no idea how to do the link task. So complicated to me. hope someone could do this to utilize the fortran code.

Subject: Re: sample/empirical variogram calculation
Posted by james-a-roo on Sun, 09 Nov 2008 16:33:08 GMT

View Forum Message <> Reply to Message

Yes and yes.

A few weeks ago, I sat through a presentation on working with sparse matrices and large arrays in R to find out, at the end of the talk, that R is pass by value. I laughed.

I wouldnt have written this code if I thought i would have been more efficient over the short term to try to link to the GSLIB fortran. However, this is something I would be interested in exploring at some later point. I'm going to be doing some such linking in the next 6 months, will attempt this for some GSLIB routines.

On Nov 6, 11:59 pm, matsu770...@gmail.com wrote:

> Maybe Gslib library is a possible choise that can be linked to IDL to
> perform some kind of geo-statistics calculation. However, it's fortran
> code library so i have no idea how to do the link task. So complicated
> to me. hope someone could do this to utilize the fortran code.

Subject: Re: sample/empirical variogram calculation
Posted by mccreigh on Thu, 20 Nov 2008 00:56:34 GMT

View Forum Message <> Reply to Message

Not surprisingly, i found some bugs. I also made some major improvements so that it's doing an "upper triangular" thing, instead of the full "matrix". (it's actually looping through said matrix, so it's not really a matrix.) Should be substantially faster. I'm currently verifying it against some collaborator's results, things look good running on random subsamples.

anyway, if you would like an updated version, let me know. I hope to
soon have web space and provide a link here. I mostly wanted to reply
right now because i found bugs.

On Nov 9, 9:33am, james-a-roo <james.mccrei...@gmail.com> wrote:
> Yes and yes.
>
> A few weeks ago, I sat through a presentation on working with sparse
> matrices and large arrays in R to find out, at the end of the talk,
> that R is pass by value. I laughed.
>
> I wouldnt have written this code if I thought i would have been more
> efficient over the short term to try to link to the GSLIB fortran.
> However, this is something I would be interested in exploring at some
> later point. I'm going to be doing some such linking in the next 6
> months, will attempt this for some GSLIB routines.
>
> On Nov 6, 11:59 pm, matsu770...@gmail.com wrote:
>
>> Maybe Gslib library is a possible choise that can be linked to IDL to
>> perform some kind of geo-statistics calculation. However, it's fortran
>> code library so i have no idea how to do the link task. So complicated
>> to me. hope someone could do this to utilize the fortran code.
>
>

Subject: Re: sample/empirical variogram calculation
Posted by zhli.gla on Mon, 30 Dec 2013 11:20:56 GMT

View Forum Message <> Reply to Message

Hi, could you please email me a copy of your updated version? Thanks!

On Thursday, 20 November 2008 00:56:34 UTC, changuito wrote:
> Not surprisingly, i found some bugs. I also made some major
> improvements so that it's doing an "upper triangular" thing, instead
> of the full "matrix". (it's actually looping through said matrix, so
> it's not really a matrix.) Should be substantially faster. I'm
> currently verifying it against some collaborator's results, things
> look good running on random subsamples.
>
> anyway, if you would like an updated version, let me know. I hope to
> soon have web space and provide a link here. I mostly wanted to reply
> right now because i found bugs.
>

>
> On Nov 9, 9:33am, james-a-roo <james.mccrei...@gmail.com> wrote:
>> Yes and yes.
>>
>> A few weeks ago, I sat through a presentation on working with sparse
>> matrices and large arrays in R to find out, at the end of the talk,
>> that R is pass by value. I laughed.
>>
>> I wouldnt have written this code if I thought i would have been more
>> efficient over the short term to try to link to the GSLIB fortran.
>> However, this is something I would be interested in exploring at some
>> later point. I'm going to be doing some such linking in the next 6
>> months, will attempt this for some GSLIB routines.
>>
>> On Nov 6, 11:59 pm, matsu770...@gmail.com wrote:
>>
>>> Maybe Gslib library is a possible choise that can be linked to IDL to
>>> perform some kind of geo-statistics calculation. However, it's fortran
>>> code library so i have no idea how to do the link task. So complicated
>>> to me. hope someone could do this to utilize the fortran code.
>>
>>